Quantum Computation (Preskill ch. 6)

Quantum Computation

Classical Circuits

- Universal Gates
- Circuit Complexity

Quantum Circuits

- Quantum Complexity
- Universal Quantum Gates

Review of Classical Circuit Theory

Think of a Computation as a function that maps n bits to m bits

$$\boxed{f: \{0,1\}^n \rightarrow \{0,1\}^m}$$

Maps n bits to m bits

A function with an m bit output is equivalent to m functions with a *one* bit output, so the basic task can be broken into m functions mapping n bits to *one* bit

There are 2^n possible inputs w/2 possible outputs, so a total of 2^{2^n} functions that map n bits to *one* bit

$$\boxed{f: \{0,1\}^n \rightarrow \{0,1\}}$$

2^{2^n} of these simple functions

$$\boxed{\text{Function evaluation} \longleftrightarrow \text{sequence of logic operations}}$$

Given a binary input $x = x_1 x_2 \ldots x_n$

⇒ separate in sets $\begin{cases} f(x) = 1 \\ f(x) = 0 \end{cases}$

Consider the input

$x^{(a)}: f(x^{(a)}) = 1$ ⇒ define $f^{(a)}(x) = \begin{cases} 1 & \text{for } x = x^{(a)} \\ 0 & \text{for } x \neq x^{(a)} \end{cases}$

one of the m simple functions

n of these

Given, for example, we implement $f^{(a)}$ w/logic operations

$$x = \begin{cases} 111\ldots \rightarrow f(x) = x_1 \wedge x_2 \wedge x_3 \ldots \wedge x_n \\ 0110\ldots \rightarrow f(x) = (\neg x_1) \wedge x_2 \wedge x_3 \wedge (\neg x_4) \ldots \end{cases}$$

Finally, given the $f^{(a)}(x)$'s we can implement the $f(x)$'s as

$$\boxed{f(x) = f^{(1)}(x) \vee f^{(2)}(x) \vee \ldots \vee f^{(n)}(x)}$$

Quantum Computation (Preskill ch. 6)

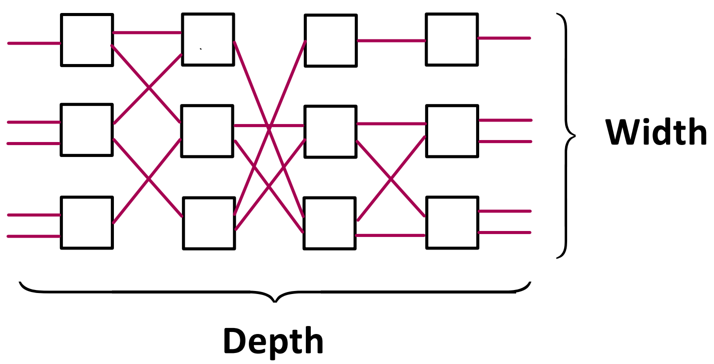
Circuit Complexity

(Pick a universal gate set)

Central Question: How hard is it to solve **PROBLEM** ?

* One measure is the size of the smallest circuit that solves it

Size = Width x Depth



Consider a circuit family $\{C_n\}$ that solves a decision problem $f: \{0,1\}^n \to \{0,1\}$

Examples

FACTORING $f(x,y) = \begin{cases} 1 & \text{if integer } x \text{ has divisor } < y \\ 0 & \text{otherwise} \end{cases}$

HAMILTONIAN PATH $f(x,y) = \begin{cases} 1 & \text{if graph } x \text{ has Hamiltonian Path} \\ 0 & \text{otherwise} \end{cases}$

We define:

Easy Problems: $\text{size}(C_n) \leq \text{poly}(n)$

Hard Problems: $\text{size}(C_n) > \text{poly}(n)$

This distinction allows us to define Complexity Classes, for example

Problem Class $P = \left\{ \text{Decision Problems solved by a polynomial-sized circuit} \right\}$

Quantum Computation (Preskill ch. 6)

Consider a circuit family $\{C_n\}$ that solves a <u>decision problem</u> $f: \{0,1\}^n \rightarrow \{0,1\}$

Examples

FACTORING $f(x,y) = \begin{cases} 1 & \text{if integer } x \text{ has divisor } < y \\ 0 & \text{otherwise} \end{cases}$

HAMILTONIAN PATH $f(x,y) = \begin{cases} 1 & \text{if graph } x \text{ has Hamiltonian Path} \\ 0 & \text{otherwise} \end{cases}$

We <u>define</u>:

Easy Problems:	$\text{size}(C_n) \leq \text{poly}(n)$
Hard Problems:	$\text{size}(C_n) > \text{poly}(n)$

This distinction allows us to define <u>Complexity Classes</u>, for example

Problem Class P = { Decision Problems solved by polynomial-sized circuit }

∗ Whether PROBLEM $\in \mathcal{P}$ is independent of circuit design, universal gate set & other specifics

∗ Problems in $\mathcal{P}$ are special – they have structure that allows efficient computation

<u>Note</u>: The majority of functions $\notin \mathcal{P}$

For example, if the output $f(x) \sim$ random we must compute $f(x)$ by lookup table with 2^n entries

Circuit that does lookup has exponential size

<u>Special Class</u>: One-Way Function

Problem Class NP = { PROBLEM is easy or hard, but the answer is easy to check }

Stands for "Non-deterministic Polynomial Time

<u>Examples</u>: FACTORING $\in$ NP

HAMILTONIAN PATH $\in$ NP

<u>Note</u>: <u>Clearly</u> $P \subseteq NP$, <u>Conjecture</u> that $P \neq NP$

Quantum Computation (Preskill ch. 6)

* Whether PROBLEM $\in P$ is independent of circuit design, universal gate set & other specifics

* Problems in P are special – they have structure that allows efficient computation

Note: The majority of functions $\notin P$

For example, if the output $f(x) \sim$ random we must compute $f(x)$ by lookup table with 2^n entries

Circuit that does lookup has exponential size

Special Class:

One-Way Function

Problem Class NP = { PROBLEM is easy or hard, but the answer is easy to check }

Stands for "Non-deterministic Polynomial Time

Examples: FACTORING $\in$ NP

HAMILTONIAN PATH $\in$ NP

Note: **Clearly** $P \subseteq NP$, **Conjecture** that $P \neq NP$

Special Problem: CIRCUIT-SAT $\in$ NP

Input = Circuit w/ n gates, m input bits

Problem = is there an m-bit input w/output = 1

$$f(C) = \begin{cases} 1 & \text{if } \exists\, x^{(m)} \text{ so } C(x^{(m)}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Easy to check solution because if we have the input circuit C we can run it with the input $x^{(m)}$ and determine if it evaluates to 1.

Cooks Theorem: Every PROBLEM $\in$ NP is polynomially reducible to CIRCUIT-SAT

$\left.\begin{array}{c}\text{CIRCUIT-SAT} \\ \text{HAMILTONIAN PATH}\end{array}\right\} \in \boxed{\text{NP- Complete}}$

NPC ≠ NP

Quantum Computation (Preskill ch. 6)

Special Problem: CIRCUIT-SAT ∈ NP

Input = Circuit w/n gates, m input bits

Problem = is there an m-bit input w/output = 1

$$f(C) = \begin{cases} 1 & \text{if } \exists\, x^{(m)} \text{ so } C(x^{(m)}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Easy to check solution because if we have the input circuit C we can run it with the input $x^{(m)}$ and determine if it evaluates to 1.

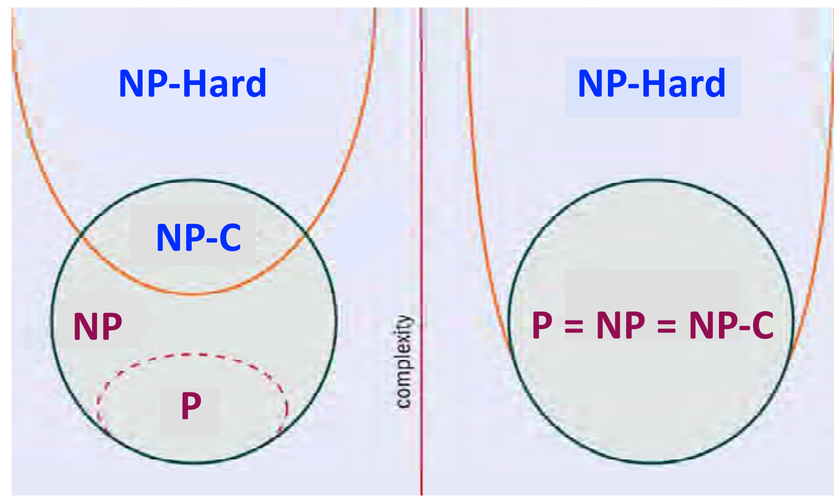
Cooks Theorem: Every PROBLEM ∈ NP is polynomially reducible to CIRCUIT-SAT

$$\left.\begin{array}{c}\text{CIRCUIT-SAT} \\ \text{HAMILTONIAN PATH}\end{array}\right\} \in \boxed{\text{NP- Complete}}$$

NPC ≠ NP

Complexity Hierarchy

- **Conjecture**: P ∈ NP
- ∃ Problems in NP that are neither P or NPC
- NPI: Problems of intermediate difficulty
- Conjecture: Factoring ∈ NPI



P ≠ NP P = NP

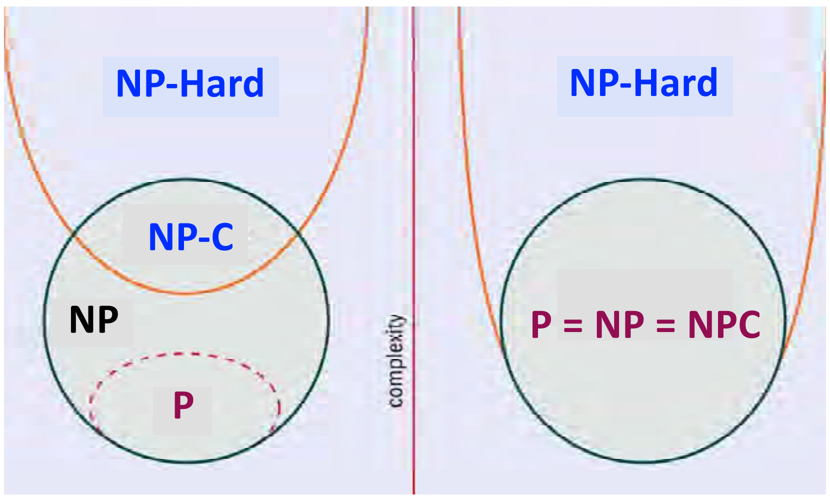
Takeaway Message

- Complexity theory is a rich field with many known complexity classes
- Many foundational conjectures remain unproven
- As we will see, switching to Quantum Circuits changes things

Quantum Computation (Preskill ch. 6)

Complexity Hierarchy

- **Conjecture**: P ∈ NP
- ∃ Problems in NP that are neither P or NPC
- NPI: Problems of intermediate difficulty
- Conjecture: Factoring ∈ NPI



P ≠ NP   P = NP

Takeaway Message

- Complexity theory is a rich field with many known complexity classes
- Many foundational conjectures remain unproven
- As we will see, switching to Quantum Circuits changes things

Aside: Classical Reversible Computation

Motivation:

Quantum Computation = Unitary Transformation

⇓

Reversible !

Classical Reversible Comp: $f: \{0,1\}^n \to \{0,1\}^n$

Repackage $f: \{0,1\}^n \to \{0,1\}^m$ as reversible

$$\boxed{\begin{array}{l} f: \{0,1\}^{n+m} \to \{0,1\}^{n+m} \\ f(x, 0^{(m)}) = (x; f(x)) \end{array}}$$

we separate $n + m$ qubit register into input and output so no information is lost

Note: Not all 1 & 2-bit gates are reversible, e. g., AND, OR, ERASE

Quantum Computation (Preskill ch. 6)

Universal Quantum Gates

* What constitutes a universal gate set ?

  Answer: Almost any generic 2–qubit quantum gate will do!

* What is a generic gate?

A *k*–qubit gate $U = 2^k \times 2^k$ matrix w/evals $\{e^{i\theta_1}, \ldots e^{i\theta_{2^k}}\}$ is generic if

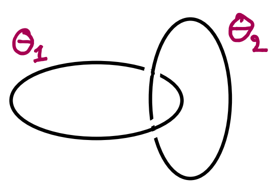
θ_i is an irrational multiple of π

θ_i, θ_j are incommensurate (θ_i / θ_j irrational multiple of π)

(1) Powers of a generic gate:

$U^n \rightarrow$ evals $\{e^{in\theta_1}, \ldots, e^{in\theta_{2^k}}\}$

points on 2^k dim torus



$\left.\begin{array}{l} U \text{ generic} \\ n \in N_0 \end{array}\right\} \Rightarrow$ points densely covers the whole torus

Definition:

Let $U = e^{iH_j dt}$ be generic (H_j is the generator of U)

$\exists n \in N_0$ so U^n comes arbitrarily close to $U(\alpha) = e^{i\alpha H_j dt}$

($U(\alpha)$ is reachable by powers U^n)

Seems extraordinarily cumbersome! Why do it that way?

Answer: This is necessary to

* Limit the gate set and and keep scaling arguments related to circuit size.
* Establish coarse graining $\rightarrow$ required for fault tolerance

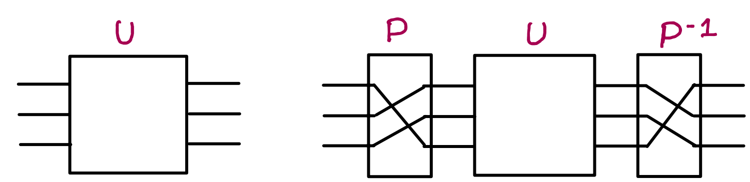
  $\{U^n, n \in N_0\}$ is a set of measure zero $\rightarrow$ any "noise takes us to an invalid state that can be detected and corrected.
* Note: It is not how current quantum circuits work!

This is not enough! What else can we do?

(2) Switching leads

k qubits $\rightarrow$ $(2^k)!$ permutations $U' = P\,U\,P^{-1}$

k = 3



This is not enough! What else can we do?

Quantum Computation (Preskill ch. 6)

Definition:

Let $U = e^{iH_j dt}$ be generic (H_j is the generator of U)

$\exists n \in N_0$ so U^n comes arbitrarily close to $U(\alpha) = e^{i\alpha H_j dt}$

($U(\alpha)$ is reachable by powers U^n)

Seems extraordinarily cumbersome! Why do it that way?

Answer: This is necessary to

* Limit the gate set and and keep scaling arguments related to circuit size.
* Establish coarse graining → required for fault tolerance

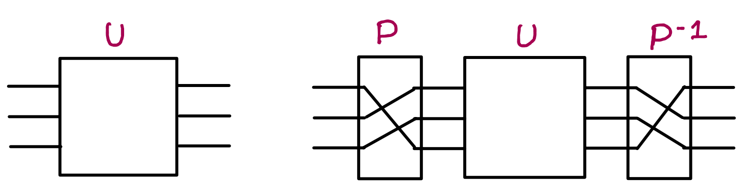
  $\{U^n, n \in N_0\}$ is a set of measure zero → any "noise takes us to an invalid state that can be detected and corrected.
* Note: It is not how current quantum circuits work!

This is not enough! What else can we do?

(2) Switching leads

k qubits → $(2^k)!$ permutations $U' = P U P^{-1}$

$k = 3$



This is not enough! What else can we do?

Aside: Consider a d - dimensional Hilbert space $\mathcal{H}$.

⇒ { Operators ($d \times d$ matrices) are vectors $\in d^2$ dim. vector space $\mathcal{H}'$ w/a scalar product defined as

$$(m_i | m_j) = \mathrm{Tr}[m_i^\dagger m_j]$$

$\exists$ orthonormal basis $\begin{cases} \{|A_1), \dots |A_{d^2})\} \\ (A_i | A_j) = \delta_{ij} \end{cases}$ in $\mathcal{H}'$

(3) Completing the Lie Algebra

Assume access to a set of Hamiltonians

$$\{H_0, H_1, \dots H_n\}, \; n \le (\dim \mathcal{H})^2$$

Trotter Formulae: for dt -> 0

$e^{-i\alpha H_j dt} e^{-i\beta H_k dt} = e^{-i(\alpha H_j + \beta H_k) dt}$ (Lin. Comb. of H_j, H_k)

$e^{-i\alpha H_j dt} e^{-i\beta H_k dt} e^{i\alpha H_j dt} e^{i\beta H_k dt} = e^{-[\alpha H_j, \beta H_k] dt^2}$ (NL. Comb. of H_j, H_k)

Quantum Computation (Preskill ch. 6)

Aside: Consider a d - dimensional Hilbert space $\mathcal{H}$.

$\Rightarrow$ $\left\{\begin{array}{l}\text{Operators } (d \times d \text{ matrices) are vectors } \in d^2 \text{ dim.} \\ \text{vector space } \mathcal{H}' \text{ w/a scalar product defined as}\end{array}\right.$

$$(m_i|m_j) = \mathrm{Tr}[m_i^\dagger m_j]$$

$\Downarrow$

$\exists$ orthonormal basis $\left\{\begin{array}{l}\{|A_1),\ldots|A_{d^2})\} \\ (A_i|A_j) = \delta_{ij}\end{array}\right.$ in $\mathcal{H}'$

(3) Completing the Lie Algebra

Assume access to a set of Hamiltonians

$$\{H_0, H_1, \ldots H_n\},\ n \leq (\dim\mathcal{H})^2$$

Trotter Formulae: for dt -> 0

$e^{-i\alpha H_j dt} e^{-i\beta H_k dt} = e^{-i(\alpha H_j + \beta H_k)dt}$ (Lin. Comb. of H_j, H_k)

$e^{-i\alpha H_j dt} e^{-i\beta H_k dt} e^{i\alpha H_j dt} e^{i\beta H_k dt} = e^{-[\alpha H_j, \beta H_k]dt^2}$
(NL. Comb. of H_j, H_k)

* From the Set $\{H_0, H_1, \ldots H_n\}$ we can "simulate" new Hamiltonians using the Trotter formulae

* If a new Hamiltonian is linearly independent we add it to the set.

* Continue until the Set has $d^2 = (\dim\mathcal{H})^2$ linearly independent members (Lie Algebra complete)*)

$\Rightarrow$ Set is a basis in $d^2 \times d^2$ matrix space
Allows to simulate any $H(t)$ & implement any U

Examples:

$d = 2 \rightarrow \{|A_i)\} = \{I, \sigma_x, \sigma_y, \sigma_z\} \leftarrow$ set of $2^2 = 4$ 2 x 2 matrices

$d = 4 \rightarrow \{|A_i)\} \leftarrow$ set of $d^2 = 16$ 4 x 4 matrices

Example: (single qubit control)

Let $d = 2$, initial set $\{\alpha\sigma_x, \beta\sigma_y\}$ (generic)

$[\sigma_x, \sigma_y] = i\sigma_z \rightarrow$ we can simulate $i\gamma\sigma_z$

*) This is not always possible. The Lie Algebra may "close" before generating a basis. If so, add more Hamiltonians to the original set.

Quantum Computation (Preskill ch. 6)

Aside: Consider a d - dimensional Hilbert space $\mathcal{H}$.

$\Rightarrow$ $\left\{ \begin{array}{l} \text{Operators } (d \times d \text{ matrices) are vectors } \in d^2 \text{ dim.} \\ \text{vector space } \mathcal{H}' \text{ w/a scalar product defined as} \end{array} \right.$

$$(m_i | m_j) = \mathrm{Tr}[m_i^{+} m_j]$$

$\Downarrow$

$\exists$ orthonormal basis $\left\{ \begin{array}{l} \{|A_1), \ldots |A_{d^2})\} \\ (A_i | A_j) = \delta_{ij} \end{array} \right.$ in $\mathcal{H}'$

(3) Completing the Lie Algebra

Assume access to a set of Hamiltonians

$$\{H_0, H_1, \ldots H_n\},\ n \leq (\dim \mathcal{H})^2$$

Trotter Formulae: for dt -> 0

$$e^{-i\alpha H_j dt} e^{-i\beta H_k dt} = e^{-i(\alpha H_j + \beta H_k)dt}$$ (Lin. Comb. of H_j, H_k)

$$e^{-i\alpha H_j dt} e^{-i\beta H_k dt} e^{i\alpha H_j dt} e^{i\beta H_k dt} = e^{-[\alpha H_j, \beta H_k] dt^2}$$ (NL. Comb. of H_j, H_k)

* From the Set $\{H_0, H_1, \ldots H_n\}$ we can "simulate" new Hamiltonians using the Trotter formulae
* If a new Hamiltonian is linearly independent we add it to the set.
* Continue until the Set has $d^2 = (\dim \mathcal{H})^2$ linearly independent members (Lie Algebra complete)*)

$\Rightarrow$ Set is a basis in $d^2 \times d^2$ matrix space
Allows to simulate any $H(t)$ & implement any U

Examples:

$d = 2 \rightarrow \{|A_i)\} = \{I, \sigma_x, \sigma_y, \sigma_z\} \leftarrow$ set of $2^2 = 4$ 2 x 2 matrices

$d = 4 \rightarrow \{|A_i)\} \leftarrow$ set of $d^2 = 16$ 4 x 4 matrices

Example: (single qubit control)

Let $d = 2$, initial set $\{\alpha\sigma_x, \beta\sigma_y\}$ (generic)

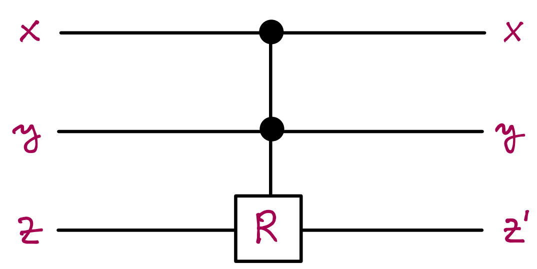
$[\sigma_x, \sigma_y] = i\sigma_z \rightarrow$ we can simulate $i\gamma\sigma_z$

Set $[I, \alpha\sigma_x, \beta\sigma_y]$ sufficient for control

Quantum Computation (Preskill ch. 6)

Deutsch's Gate

(First generic gate, Reaches any $U \in SU(8)$)



Rotation $R = -iR_x(\theta) = -ie^{i\theta/2\,\sigma_x} = -i\left(\cos\frac{\theta}{2} + i\sigma_x \sin\frac{\theta}{2}\right)$ iff $xy = 1$

incommensurate w/π

Special case $\theta = \pi$ → this is a **Toffoli gate**: $-iR_x(\pi) = -i\sigma_x$

to within a phase

Note: $R^{4n} = R_x(4n\theta)$ (b/c $i^4 = 1$) →

$$R^{(4n+1)} = (-i)\left[\cos\frac{(4n+1)\theta}{2} + i\sigma_x \sin\frac{(4n+1)\theta}{2}\right] \simeq \sigma_x \text{ for some } n$$

Action on the basis states: $R^{(4n+1)}$ transposes (6) & (7)

(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	$*$)
$\{\|000\rangle,$	$\|001\rangle,$	$\|010\rangle,$	$\|011\rangle,$	$\|100\rangle,$	$\|101\rangle,$	$\|110\rangle,$	$\|111\rangle\}$	
						$\|111\rangle,$	$\|110\rangle$	

σ_x on 2D subspace

Note: A Deutsch gate on a 3-qubit state can be cast as an 8 x 8 matrix acting in an 8-dimensional vector space.

With the basis states numbered as in $*$) above, $R^{(4n+1)}$ has the matrix representation

$$(\sigma_x)_{67} = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & \sigma_x \end{array}\right)$$

← flips the spin of the 2-level system (6),(7)

By switching leads and applying Toffoli gates, we can do any permutation of basis states. Thus we can reach

$$\boxed{P(\sigma_x)_{67}P^{-1} = (\sigma_x)_{nm}}$$

On matrix form:

Identity

$$[(\sigma_x)_{56}, (\sigma_x)_{67}] = \left[\left(\begin{array}{c|ccc} I & & 0 & \\ \hline & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 \end{array}\right), \left(\begin{array}{c|ccc} I & & 0 & \\ \hline & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ & 0 & 1 & 0 \end{array}\right)\right] = \left(\begin{array}{c|ccc} I & & 0 & \\ \hline & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ & -1 & 0 & 0 \end{array}\right) = i(\sigma_y)_{57}$$

In turn, this allows us to reach $e^{i(\sigma_x)_{56}}$ and $e^{i(\sigma_x)_{67}}$

→ we can reach $e^{-i[(\sigma_x)_{56}, (\sigma_x)_{67}]}$

Thus: $\boxed{\text{Compositions of } (\sigma_x)_{nm}\text{'s} \rightarrow i(\sigma_y)_{pq}\text{'s}}$

these generators these generators

Quantum Computation (Preskill ch. 6)

Note: A Deutsch gate on a 3-qubit state can be cast as an 8 x 8 matrix acting in an 8-dimensional vector space.

With the basis states numbered as in $*$) above, $R^{(4n+1)}$ has the matrix representation

$$(\sigma_x)_{67} = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & \sigma_x \end{array}\right) \leftarrow$$

flips the spin of the 2-level system (6),(7)

By switching leads and applying Toffoli gates, we can do any permutation of basis states. Thus we can reach

$$\boxed{P(\sigma_x)_{67}P^{-1} = (\sigma_x)_{nm}}$$

On matrix form:

Identity

$$[(\sigma_x)_{56},(\sigma_x)_{67}] = \left[\left(\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \end{array}\right),\left(\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{matrix} \end{array}\right)\right] = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{matrix} \end{array}\right) = i(\sigma_y)_{57}$$

In turn, this allows us to reach $e^{i(\sigma_x)_{56}}$ and $e^{i(\sigma_x)_{67}}$

$\rightarrow$ we can reach $e^{-i[(\sigma_x)_{56},(\sigma_x)_{67}]}$

Thus: $\boxed{\text{Compositions of } (\sigma_x)_{nm}\text{'s} \rightarrow i(\sigma_y)_{pq}\text{'s}}$

these generators

these generators

Similarly, $[(\sigma_x)_{nm},(\sigma_y)_{nm}] = i(\sigma_z)_{nm}$ $\Rightarrow$

$$\boxed{\text{Compositions of } (\sigma_x)_{nm}\text{'s} \ \& \ (\sigma_y)_{nm}\text{'s} \rightarrow (\sigma_z)_{nm}\text{'s}}$$

Conclusion: We can reach all transformations generated by linear combinations of the $(\sigma_{x,y,z})_{nm}$'s, which together span the SU(8) Lie Algebra

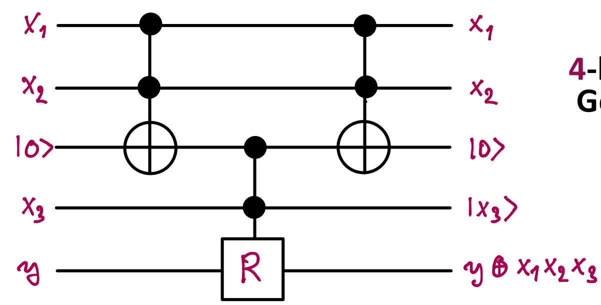
Quantum Computation (Preskill ch. 6)

<u>Similarly,</u> $[(\sigma_x)_{nm}, (\sigma_y)_{nm}] = i(\sigma_z)_{nm}$ ➡

Compositions of $(\sigma_x)_{nm}$'s & $(\sigma_y)_{nm}$'s $\rightarrow$ $(\sigma_z)_{nm}$'s

<u>Conclusion</u>: We can reach all transformations generated by linear combinations of the $(\sigma_{x,y,z})_{nm}$'s, which together span the SU(8) Lie Algebra

<u>Extending</u> to *n* bit Deutsch gate:



4-bit Deutsch Gate Generates SU(2^4)

<u>Repeat</u> $\rightarrow$ *n* bit Deutsch gate generates SU(2^n)

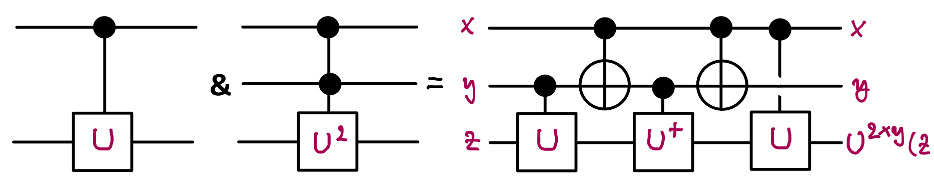
➡ The Deutsch Gate is Universal

<u>Universal 2-qubit gate sets</u>

<u>Proof</u>: can build a Deutsch gate from 2-qubit gates

<u>Assume</u> we have

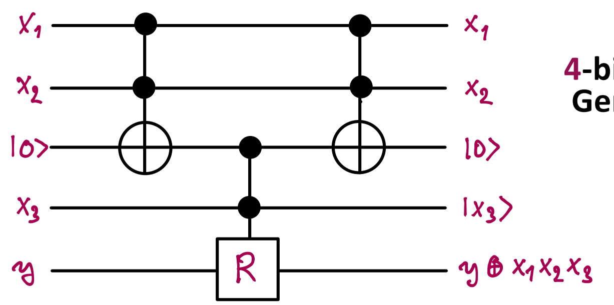
Then <u>we can do</u>



➡ We can build a Deutsch gate from $\left\{ \begin{array}{l} \text{controlled } U \\ \text{controlled } U^{-1} \\ \text{controlled NOT} \end{array} \right.$

Quantum Computation (Preskill ch. 6)

Extending to *n* bit Deutsch gate:



4-bit Deutsch Gate Generates $SU(2^4)$

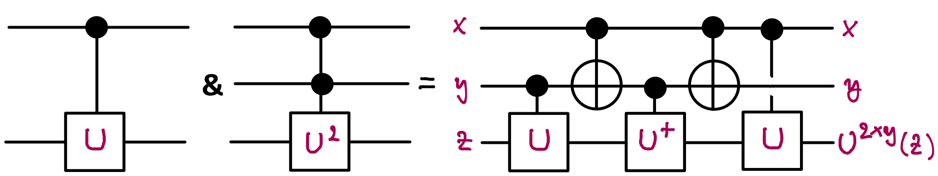
Repeat → *n* bit Deutsch gate generates $SU(2^n)$

➡ **The Deutsch Gate is Universal**

Universal 2-qubit gate sets

Proof: can build a Deutsch gate from 2-qubit gates

Assume we have — Then we can do



➡ We can build a Deutsch gate from
- controlled U
- controlled U^{-1}
- controlled NOT

where $U^2 = -iR_x(\theta)$ → can choose $U = e^{i\pi/4} R_x(\frac{\theta}{2})$

Powers of U → σ_x, U^{-1} → can build a Deutsch gate from U alone if θ/π is irrational

Generic 2-qubit gates:

Can show that any 2-qubit gate of the type

$$U = e^{iA}, \quad A = (\alpha I + \beta\sigma_x + \gamma\sigma_y)_{nm}$$

pair of states in 2-qubit $\mathcal{H}$

is universal — incommensurate α, β, γ

Other adequate sets:

Classical multi-bit gates + generic single qubit gate

e. g.:	CNOT	+	Rotations $\in SU(2)$
	Toffoli	+	$\pi/2$ Rotations

Comment on Circuit Complexity:

We still need to show that we can build a circuit that implements W to within $\mathcal{E}$ of U with #of gates $= poly(\mathcal{E}^{-1})$

Distance measure $\mathcal{E} = \|(U-W)|\psi\rangle\|_{\sup|\psi\rangle}$ ($\exists$ other measures)

This can be proved: A Quantum Computer built w/universal gates can simulate any Quantum Computer with polynomial slowdown

Quantum Computation (Preskill ch. 6)